2024 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
DIGITAL ENGINEERING/SYSTEMS ENGINEERING TECHNICAL SESSION
AUGUST 13-15, 2024 - NOVI, MICHIGAN

Synchronizing MBSE Models and Software Development in Robotic Autonomous Systems

Calvin Cheung[1], Andrew Pfeil[1], Shannon Griffith[1], Mark Petrotta[2], Mitchell Brooks[2], Michael Moore[3], Doyle Dennis[3], Reagan Grunwald[3]

[1] U.S. Army DEVCOM Ground Vehicle Systems Center, Warren, MI
[2] System Strategy Inc., Detroit, MI
[3] Moore Integrity Engineering, San Antonio, TX

ABSTRACT

MBSE and Agile software development are both vital approaches used by the DoD to facilitate development at reduced cost and a rapid pace for high priority efforts, such as Robotic and Autonomous Systems. Despite the common goals however, the expertise needed to apply both approaches are often separated between different groups of personnel with different skillsets, such as systems engineers and software developers. To bridge the gap between MBSE and Agile software development, we developed toolchains that help synchronize software development with SysML models. These toolchains leverage an XML-based, developer-maintained Model Import File (MIF) schema that we created. The MIFs are based upon Robot Operating System concepts and can be used to create additional toolchains in the future based on program needs.

Citation: C. Cheung, A. Pfeil, S. Griffith, M. Petrotta, M. Brooks, M. Moore, D. Dennis, R. Grunwald, "Synchronizing MBSE Models and Software Development in Robotic Autonomous Systems," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2024.

1. INTRODUCTION

The integration of Robotic and Autonomous Systems (RAS) with U.S. Army forces is key to improving combat effectiveness and ensuring overmatch against enemies. This notion was elaborated upon in the "Robotics and Autonomous Systems Strategy" [1], which detailed how the Army would integrate RAS technologies to meet Joint Chief of Staff goals of increasing operational options in near, mid, and far-term horizons. The importance of this human machine integration is made evident when reviewing investments that the Army is making in the RAS ground systems space. In FY2023 alone, major Army Research and Development program funding for RAS ground systems reached approximately $1.4B [2]. Given the tremendous investment and overall importance of RAS ground vehicles to Army strategy, ensuring high

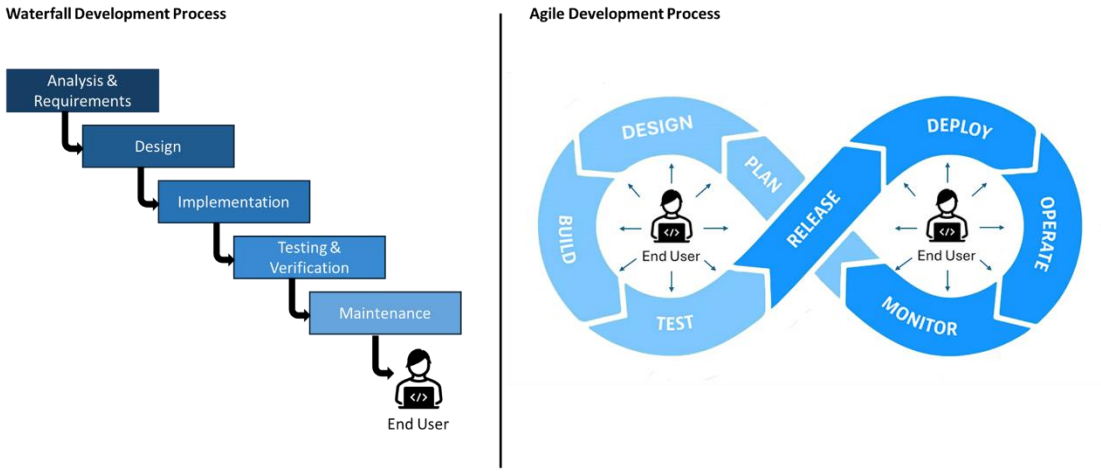

Figure 1. Comparison of Waterfall and Agile Software Development Methodologies (recreated from **[6]**).

quality outcomes in the development of these complex and costly systems is vital, which makes the application of a Model-Based Systems Engineering (MBSE) approach ideal for RAS.

MBSE is "the formalized application of modeling to support system requirements, design, analysis, and verification and validation activities beginning in the conceptual design phase and continuing throughout development and later lifecycle phases [3]." The benefits of it are myriad, including greater consistency, improved communication, design reuse, and a complete thread from requirements to implementation. Utilization of MBSE has shown to increase rates of performance and success, along with showing a reduction in defects [3]. In addition to the positive outcomes, utilization of MBSE is explicitly referenced in DoD strategy and instructions documents, such as the "DoD Digital Engineering Strategy [4]" and "DoD Instruction 5000.88 Engineering of Defense Systems [5]". Despite the benefits and guidance on the utilization of MBSE however, there remains a reluctance from developers of RAS software in fully embracing an MBSE approach.

Agile software development is a modern development methodology that "relies on flexible requirements, regular user engagement, and an understanding of the value of what has been delivered [6]." Originating from the "Manifesto for Agile Software Development [7]," Agile software development values:

- human interactions over processes
- functioning software over thorough documentation
- collaboration with customers over contract negotiation
- flexibility to respond to changes over rigid adherence to a plan

These set of values put the methodology in contrast to the traditional government software waterfall development approach, which has a much greater focus on documentation and gated steps. The differences between the two methodologies are highlighted in Figure 1.

Despite the stark differences between the traditional waterfall approach and Agile, the DoD recognizes the value that Agile development brings with its ability to rapidly develop and deploy software. The U.S. Government Accountability Office (GAO) released a report recommending the incorporation of Agile principles in both research and acquisition efforts within the DoD [6]. Previous to that report, the DoD had already released an Adaptative Acquisition Pathway process focused on Software

Acquisition that requires the use of modern iterative software development methodologies, with Agile development being named as an explicit example [8]. The benefits offered by Agile, and the recommendation of its adaptation by DoD guidance, puts it on the forefront of development methodologies to use for software heavy systems, such as RAS ground vehicles.

In the following sections, we will explore some challenges that exist in integrating MBSE with Agile, along with providing a brief introduction to RAS software development. From there, we will introduce toolchains that we have created to allow for the generation of MBSE diagrams via machine readable markup files that developers own and maintain as a part of development. The overall intent of this effort is to bridge the gap between RAS software development and MBSE, easing the transition from code to diagrams in models to enable greater adaptation of MBSE and all the associated benefits.

2. MBSE/AGILE CHALLENGES

While both MBSE and Agile are recommended by the DoD, there have been challenges in adopting both methodologies to software development efforts in a unified fashion. Table 1 shows some of the common challenges with implementing MBSE, gathered from industry surveys and MBSE professional experiences. A common theme amongst many of these challenges is the difficulty in getting developers access to the proper support, tooling, or processes for MBSE in such a way that does not make them resistant to implementing things that appear to be non-value-added effort from a software developer's perspective.

To meet the challenge of bridging the gap between MBSE and Agile in the RAS ground vehicle space, we propose the utilization of Extensible Markup Language (XML) files tailored for the development of autonomous software. These files, which are maintained by the software developers, are machine readable and can be utilized in SysML toolchains to automate the creation and importing of blocks and diagrams related to RAS ground vehicle systems into SysML models. By utilizing these files, henceforth referred to as Model Import Files (MIF), and creating RAS specific toolchains, we are able to synchronize MBSE models with software development, reducing the impact to developers while still obtaining the necessary information.

Table 1. Challenges with MBSE.

Challenge	Citation
Culture change needed to implement MBSE	[12] [13] [14] [15]
Leadership buy-in of MBSE	[12] [14]
Building and retaining modeling talent	[12] [14]
Lack of MBSE training	[12] [13] [14]
Using models to communicate with stakeholders that do not have MBSE experience	[12] [14]
Integration of modeling tools with agile project management and software development tools	[12] [13] [15]
MBSE processes considered extra work	[13] [14]
Not using Agile development as an excuse to bypass MBSE process	[13]

3. MODEL IMPORT FILE

To bridge the gap between MBSE and Agile, we utilize machine readable MIFs to synchronize between RAS software development and SysML models. Foundational to the MIFs is the Robot Operating System (ROS). ROS is framework for the development of RAS software, leveraging a collection of tools, libraries, and

conventions to enable collaborative software development and reuse [9]. It is one of the most widely used robotic software development frameworks, with approximately 550 million ROS "packages" (a unit of software organization in ROS that can contain executables, libraries, and configuration files) downloaded in 2023 [10]. By leveraging the files and syntax intrinsic to ROS, we were able to create MIF schema definitions to be used in the generation of SysML models.

3.1. ROS NOMENCLATURE

The fundamental ROS concepts [9] utilized by the MIF are listed and defined in Table 2. ROS Nomenclature

Table 2. ROS Nomenclature

Concept	Definition
Nodes	A process that performs computation, synonymous with "software module".
Messages	Strictly typed data structures that nodes pass to each other to communicate.
Topics	Named buses that nodes send messages on with a publish/subscribe message pattern.
Packages	Unit of organization in ROS software that provides all the files needed for some functionality (ROS nodes, a ROS-independent library, configuration files, etc.) for a logically standalone purpose.
Services	A reply/request process for synchronous transactions, analogous to a web service.
Parameters	A shared variable for ROS nodes stored on a parameter server.

With these ROS concepts as a starting point, we defined the MIF schema definition that would allow us to automate generation of SysML models.

3.2. MIF SCHEMA DEFINITION

The structure of the MIFs are defined by an XML schema definition (XSD) file. At its most basic level, an XSD defines what elements and attributes are allowed in an XML file, along with the proper data types [11]. The XSD for the MIFs captures important ROS interface concepts, such as nodes, messages, topics, services, and parameters. The content of the MIFs are filled out accordingly for every ROS package, and are stored in a folder along with the rest of the package files. Refer to Figure 2 for a sample XSD file, showing nested elements, attributes, and data types.

Figure 2. Sample XSD File.

3.3. MIF SYNCHRONIZATION

One of the primary reasons the XML format was chosen was due to developer familiarity with XML concepts and the ease at which developers could work with XML files. The plain text, rules-based nature of XML makes them simple for developers to open and modify as needed in their development environment of choice. This simplicity allowed us to implement processes that require the developers to maintain the information in the MIFs alongside with ROS package updates, enabling synchronization.

4. MIF/SYSML IMPORT TOOLCHAINS

The utilization of MBSE in program development is idiosyncratic, with different types of SysML modeling needs being unique to each program. By creating a consistent set of machine-readable MIFs that are synched to RAS software development, we have provided a baseline of information for different SysML toolchains to be developed according to those needs. This flexibility allows for RAS software development to be synchronized to various modeling needs of different programs in different ways, despite only requiring one set of files. Two specific use cases we have used the MIFs for are the Army Robotic Common Software (ARCS) Library Documentation Toolchain and the Autonomous Ground Vehicle Reference Architecture (AGVRA) Interface Model Library (IML) - ARCS Toolchain.

4.1. ARCS DOCS TOOLCHAIN

The ARCS Library is developed using ROS2, the newest iteration of ROS, and enables delivery of modular RAS software to the Soldier while promoting a Modular Open Systems Approach (MOSA). In the past, software architecture diagrams for ARCS Library efforts were created by hand, using diagramming/presentation software such as Microsoft PowerPoint, Microsoft Visio, or Diagrams.net. These diagrams were generalized, ignoring many details in the architecture, and were often outdated due to the manual effort needed to generate updates. Figure 3 shows an example of a manually generated RAS diagram.

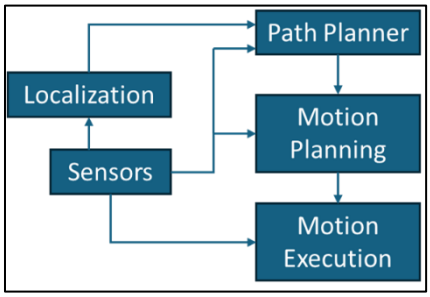

Figure 3. Example hand generated RAS software architecture diagram.

To reduce the burden of manual diagram generation, we developed an ARCS Library Documentation Toolchain. This toolchain leveraged the MIFs to automatically generate internal block diagrams of the software architecture, reducing diagraming effort and ensuring synchronicity with the codebase. Figure 4 shows an example of an internal block diagram created with SysML that could be generated from the toolchain.

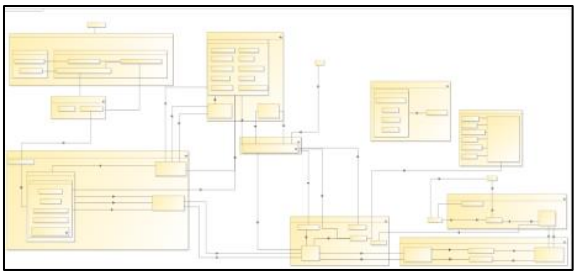

Figure 4. Example RAS software architecture diagram in SysML.

In addition to the benefits of synchronized diagram generation, utilizing the MIFs allowed the toolchain to import ROS concepts such as packages, nodes, and topics into SysML as blocks to be used for further MBSE purposes. Requirements, behavior diagrams, state diagrams, and many other

aspects would be able to be associated with and traced back to elements that were imported directly from MIFs maintained by developers, providing a consistent thread throughout the system.

4.2. AGVRA IML-ARCS TOOLCHAIN

The AGVRA IML profile defines stereotypes for describing logical and physical aspects of design elements. The logical level interface definitions relate the data used in the interfaces to the semantic datatypes defined by the AGVRA Data Model Framework. The physical level interface definitions define the storage type, encoding, and the implementation details of the interfaces. Mapping models are used to show how the logical level data interfaces are realized to physical level message interface realizations. An example IML mapping diagram can be seen in Figure 5.

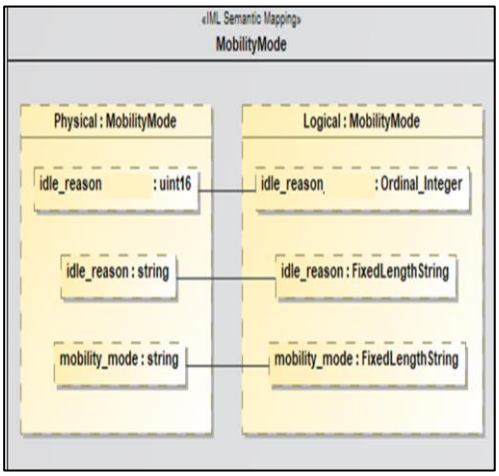

Figure 5. Example IML diagram.

To support the IML-ARCS Library toolchain, the MIF schema description includes elements for developers to enter semantic datatypes. These datatypes, defined in the AGVRA Joint Common DataType Library, and allow for the generation of IML services, ports, interfaces, and datasets in structure diagrams.

Utilization of the IML-ARCS Library diagrams provides a significant value in facilitating interoperability between the RAS software with other systems that use the IML. When data elements used in the interface definitions across systems are tied to the same semantic datatype definitions provided by the AGVRA JCDL, the data are inherently related through the JCDL datatype referenced. The semantic similarities and differences between a dataset in the ARCS Library and the same concept defined in a different system can be discovered by tracing the semantic datatypes. The need for transformations can be automatically identified in the modeling tools. All this leads to greater coherence, modularity and interoperability between RAS software and other systems.

5. CONCLUSION

MBSE and Agile software development are both important priorities of the DoD to facilitate the development of high-quality systems at reduced cost and a rapid pace. Despite the desire to be focused on both areas, there are elements of MBSE and Agile software development that are rife for conflict without the right culture shift and processes overall. In this work, we presented our utilization of MIFs to bridge the gap between the MBSE processes and Agile developer flexibility in RAS software development. We have demonstrated that MIFs are a flexible tool that can be used to develop a variety of frameworks to help meet MBSE needs while keeping synchronized with software development. By continuing down this path of tool development for automated synchronization, we look to expand the integration between Agile software development and MBSE to increase both quality of systems and speed of development without sacrifice, resulting in the robust and rapidly fielded RAS systems to support the Soldiers.

6. REFERENCES

[1] Army Capabilities Integration Center, "The U.S. Army Robotic and Autonomous Systems Strategy," U.S. Army Training and Doctrine Command, Fort Eustis, VA, 2017.

[2] B. Leggieri, "The Army's Autonomous Systems – The Ultimate Force Multiplier," Federal Budget IQ, 03 07 2023. [Online]. Available: https://federalbudgetiq.com/insights/the-armys-autonomous-systems-the-ultimate-force-multiplier/. [Accessed 06 02 2024].

[3] E. R. Carroll and R. J. Malins, "Systematic Literature Review: How is Model-Based Systems Engineering Justified?," Sandia National Laboratories, Albuquerque, NM, 2016.

[4] Office of the Deputy Assistant Secretary of Defense for Systems Engineering, "Department of Defense Digital Engineering Strategy," Office of the Deputy Assistant Secretary, Washington, DC, 2018.

[5] Office of the Under Secretary of Defense for Research and Engineering, *DoD Instruction 5000.88 Engineering of Defense Systems*, Department of Defense: Washington, DC, 2020.

[6] United States Government Accountability Office, "DEFENSE SOFTWARE ACQUISITIONS Changes to Requirements, Oversight, and Tools Needed for Weapon Programs," United States Government Accountability Office, Washington, DC, 2023.

[7] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland and D. Thomas, "Manifesto for Agile Software Development," 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed 9 02 2024].

[8] Office of the Under Secretary of Defense for Acquisition and Sustainment, "DOD Instruction 5000.87 Operation of the Software Acquisition Pathway," Office of the Under Secretary of Defense for Acquisition and Sustainment, Washington, DC, 2020.

[9] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source*, Kobe, Japan, 2009.

[10] K. Scott and T. Foote, "2023 ROS Metrics Report," Open Robotics, Mountain View, CA, 2023.

[11] W3Schools, "XML Schema Tutorial," W3Schools, [Online]. Available: https://www.w3schools.com/xml/schema_intro.asp. [Accessed 13 2 2024].

[12] M. Pantano and F. Galiber III, "Applying an Agile Approach with MBSE," in *Agile in Government Summit*, Washington, DC, 2019.

[13] M. Hause, "How to Fail at MBSE," Atego, 2013.

[14] M. Chami and J.-M. Bruel, "A Survey on MBSE Adoption Challenges," in *The Systems Engineering Conference of the Europe, Middle-East and Africa (EMEA) Sector of INCOSE*, Berlin, Germany, 2018.

[15] J. Kößler and K. Paetzold, "Integration of MBSE into existing development processes - Expectations and challenges," in *DS 87-3 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 3: Product, Services and Systems Design*, Vancouver, Canada, 2017.